

OPERATION MANUAL

Fieldbus-Controller 9251 EtherCAT Integration into TwinCAT

© 2021 burster
präzisionsmesstechnik gmbh & co kg
All rights reserved

Manufacturer:
burster
präzisionsmesstechnik gmbh & co kg
Talstr. 1 - 5 P.O. Box 1432
76593 Gernsbach 76587 Gernsbach
Germany Germany

Valid from: **01.04.2021**
Applies to: **9251-V1X00**

Tel.: +49-7224-645-0
Fax.: +49-7224-645-88
Email: info@burster.com
www.burster.com

4077-BA9251ETHERCATEN-5999-041525

Table of Contents

1	Introduction.....	3
2	Creating new project.....	4
3	Installation of ESI description files.....	6
4	Scan EtherCAT devices	7
5	Create a sample program	10
6	Further Examples	16
6.1	Read and Write of 'real' data types	16
6.2	Read and Write of 'string' data types	20

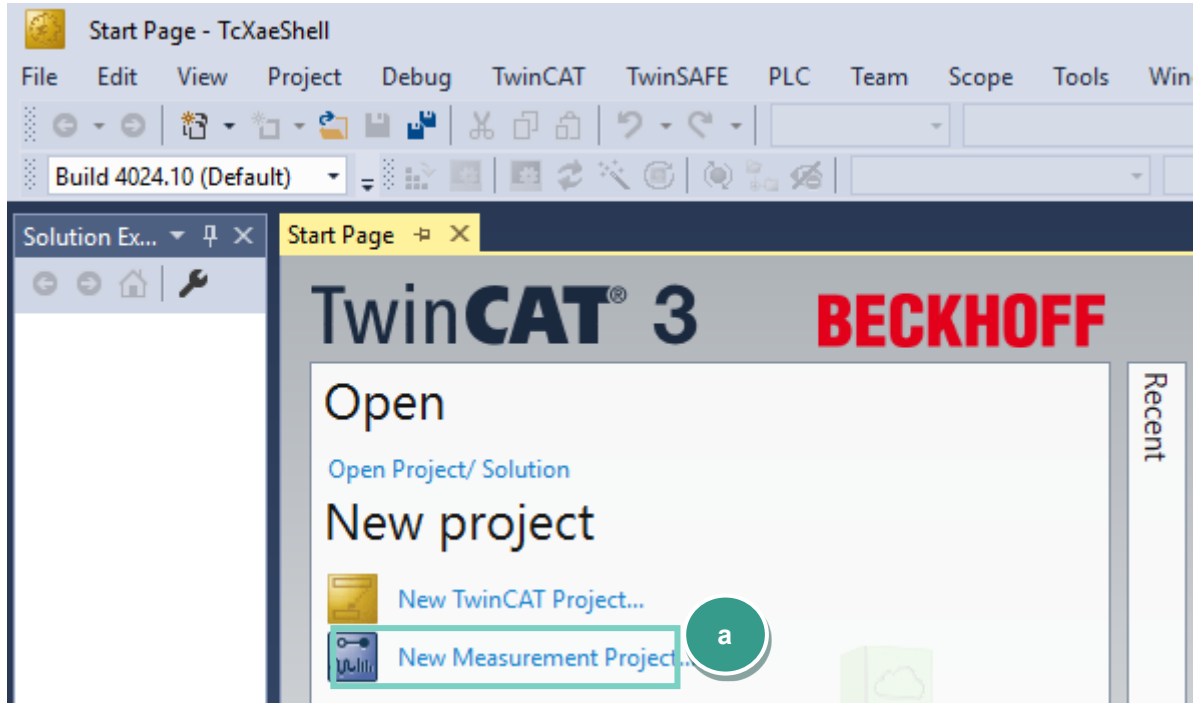
1 Introduction

This quick start guide describes an approach how you can configure the 9251 via Beckhoff TwinCAT using a Beckhoff PCI-Ethernet Card. Please note that the samples here cannot be directly used in your production line because they have been extremely simplified to reach a better understanding. Therefore, you may have to complete them by checking of status, error, length values etc.

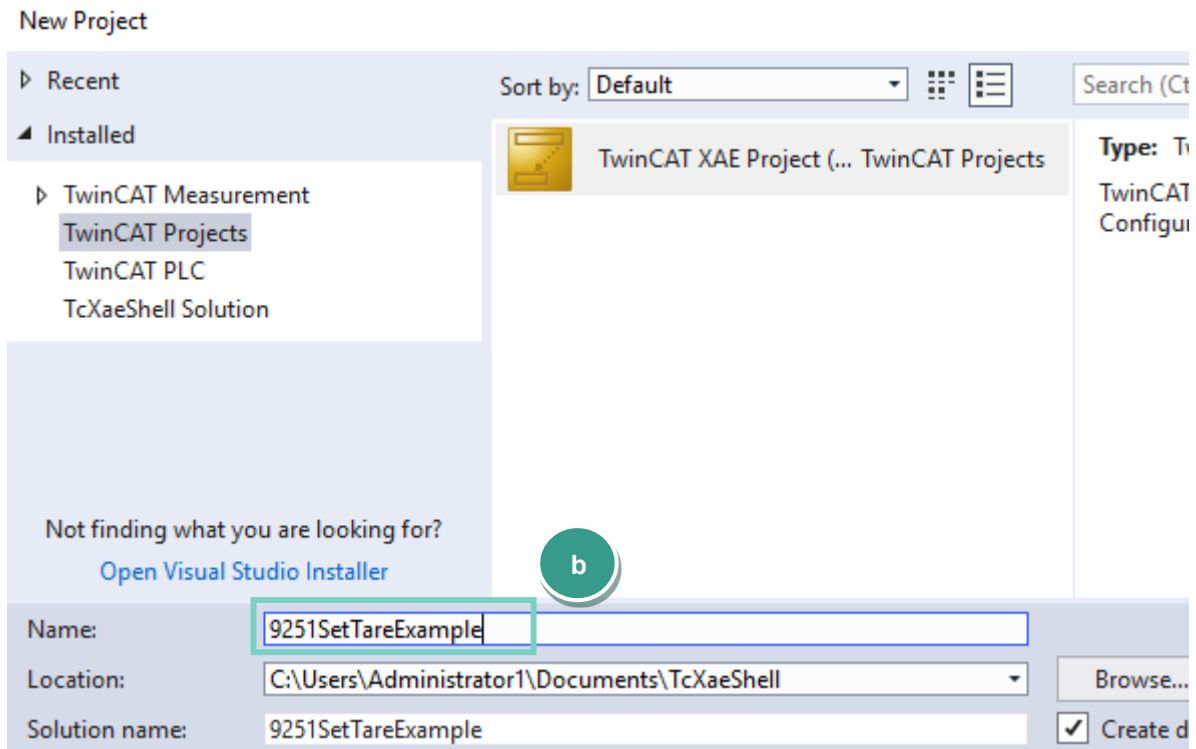
Please also note that you will have to use the 9251 manual to get further information about input and output parameters (cyclic as well acyclic data transfer)

2 Creating new project

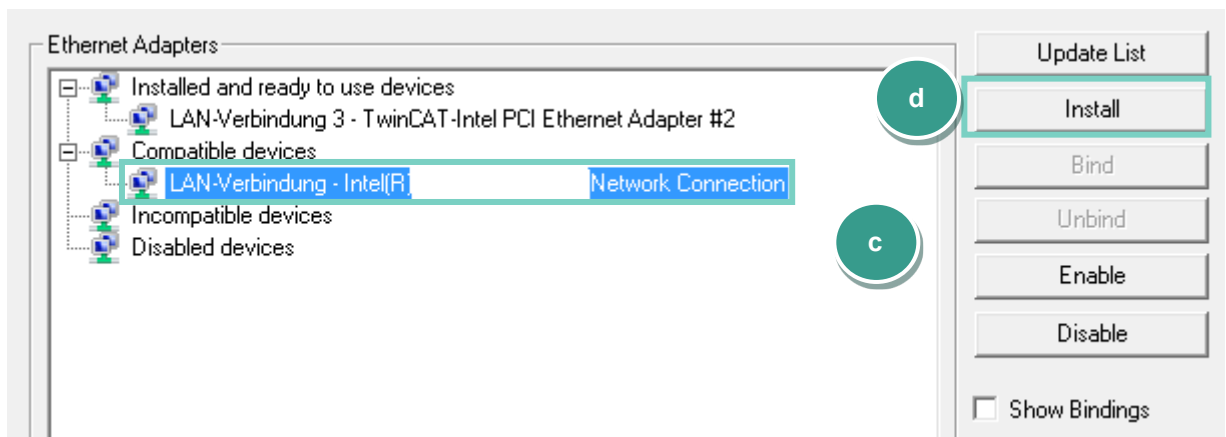
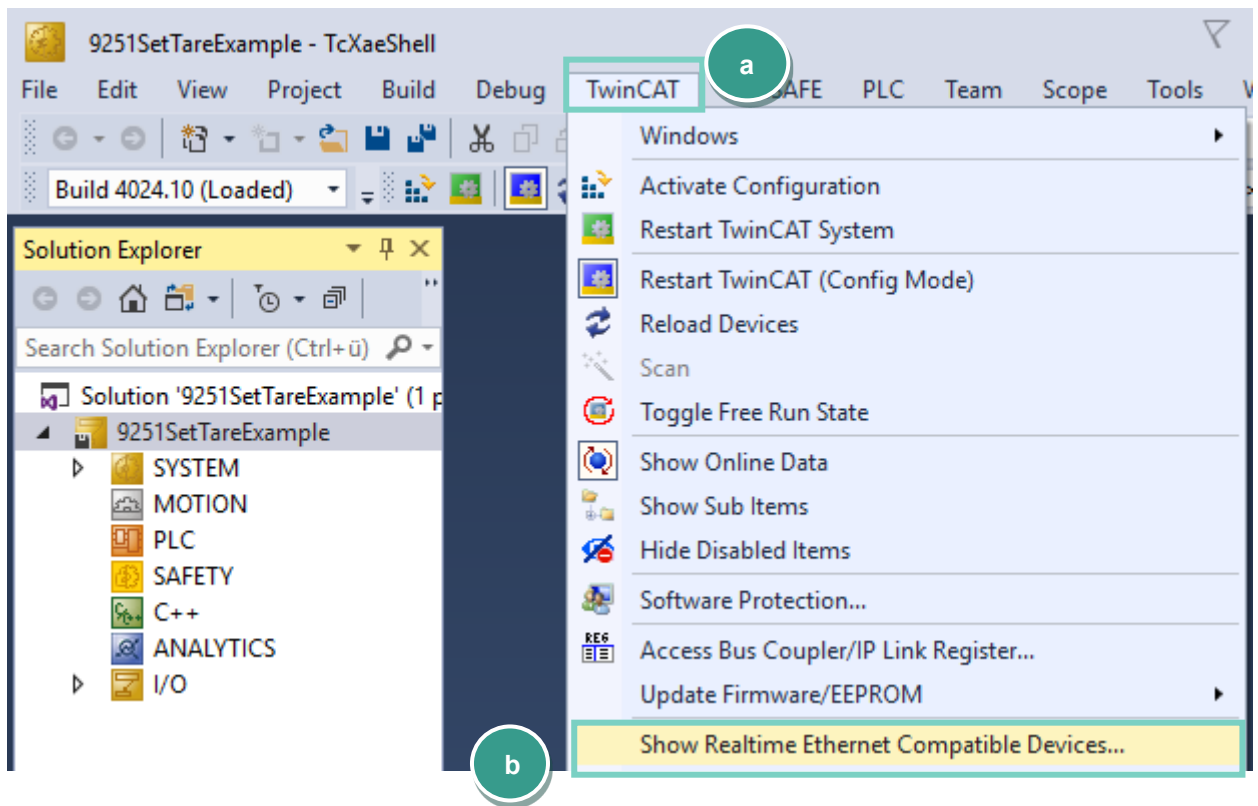
- Start the TwinCAT XAE Shell and click on *New TwinCAT Project* (or via *File* → *New Project*)



- Select *TwinCAT XAE Project*, assign a project a name (b) and click **OK**



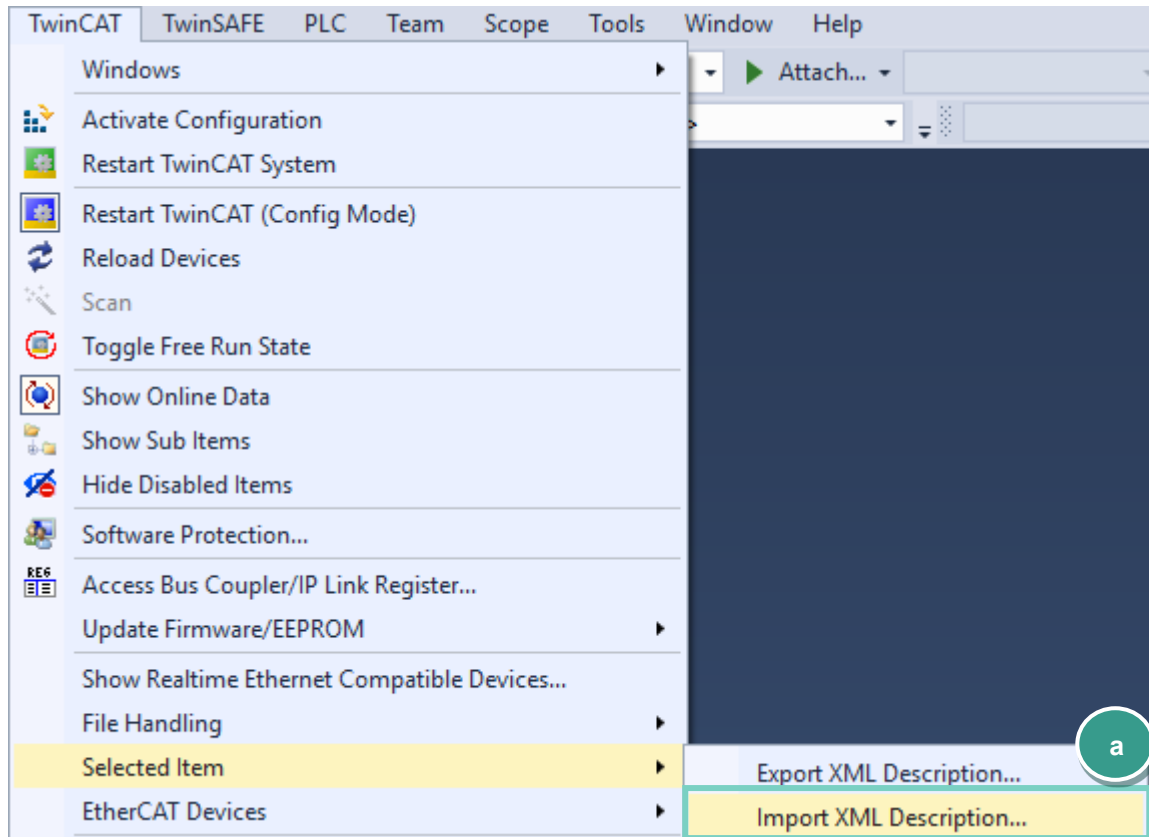
- Go to **TwinCAT** (a), select **Show Real Time Ethernet Compatible Devices...** (b) and look for you're a EtherCAT Master device under Compatible devices (c). Afterwards click the **Install** button (d).



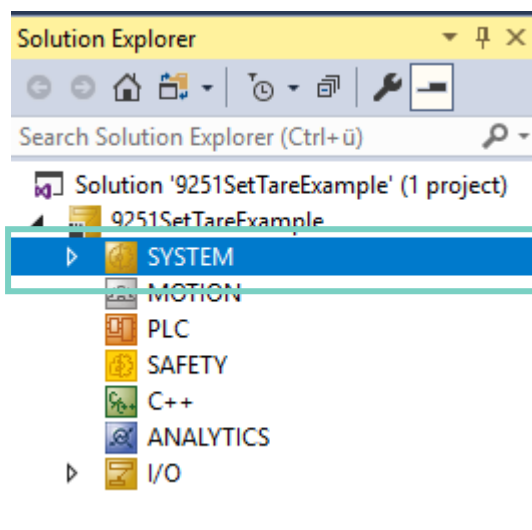
3 Installation of ESI description files

NOTE: Please make sure that your ESI file is compatible to the field bus firmware in the 9251.

- Go to TwinCAT → Selected Item → Import XML Description (a)



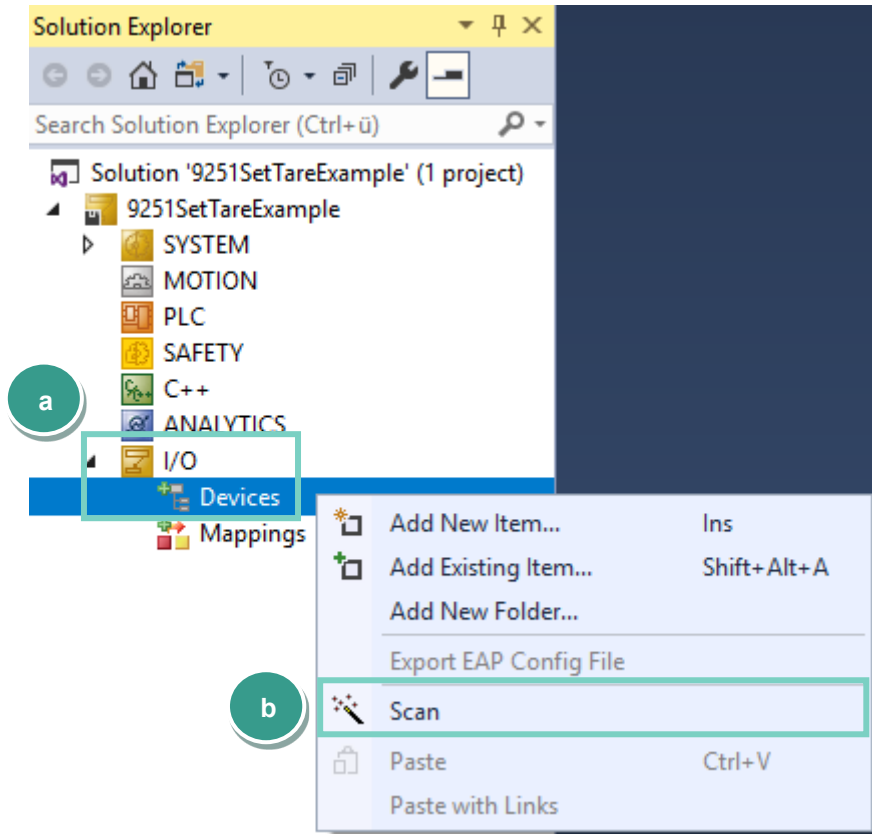
NOTE: This function is only available if the menu item SYSTEM in the project tree is selected:



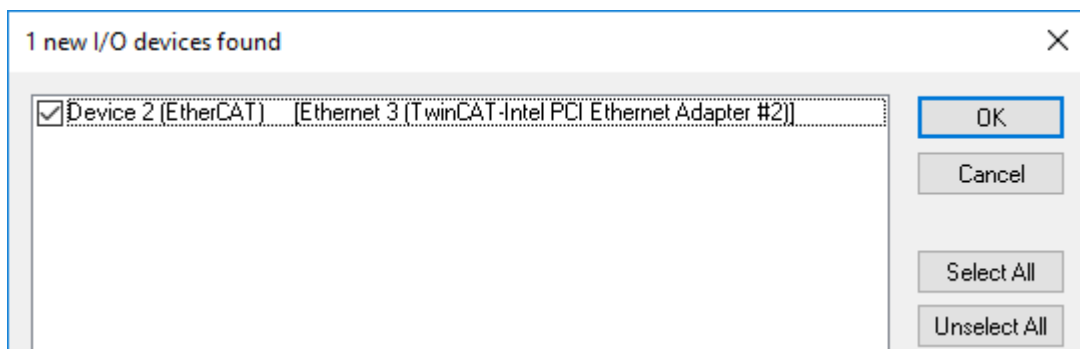
- Navigate to your 9251 ESI directory (you will find the ESI files on <https://www.burster.com/en/download-area>), select the GSD file and click **Open**.

4 Scan EtherCAT devices

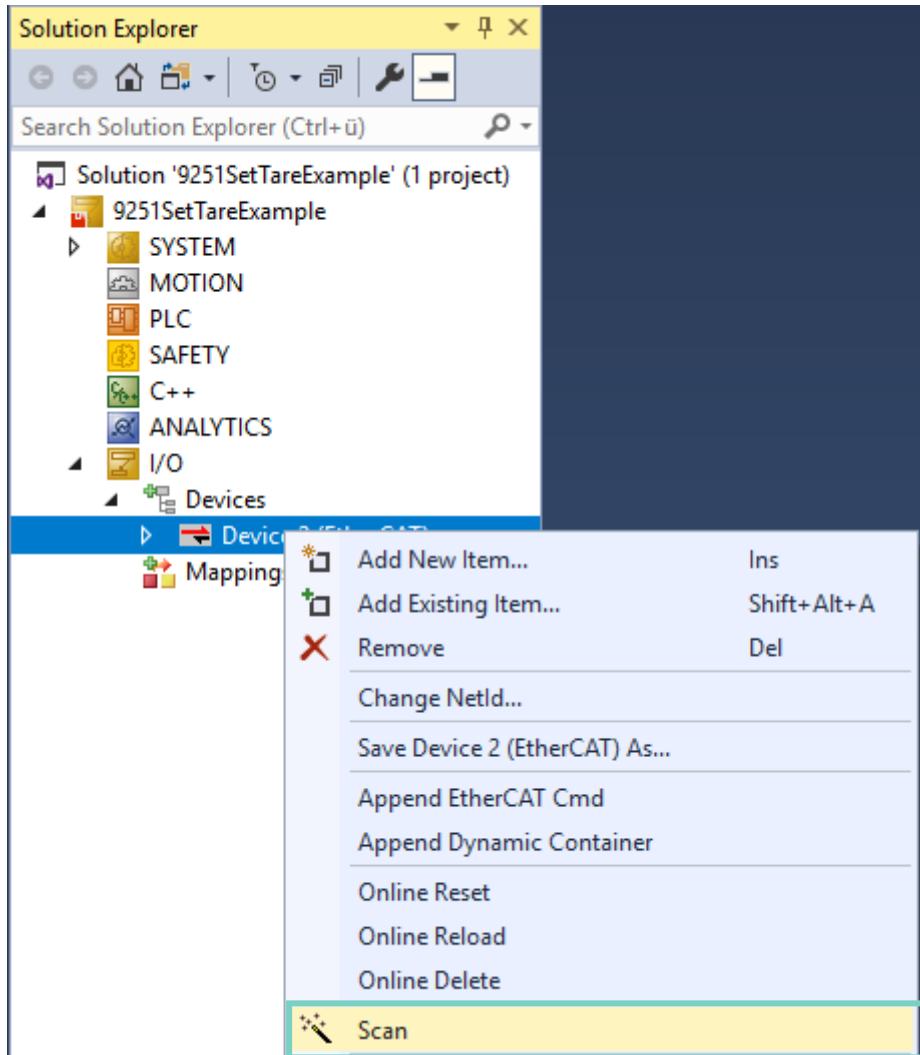
- Right click **I/O** → **Devices** (a) in the project tree und select **Scan** (b):



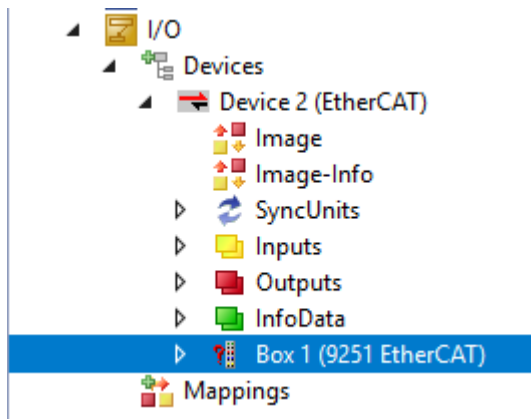
- Now, you can select an EtherCAT compatible device in the new window and click OK:



- At this point you are ready to connect the 9251 to your EtherCAT master and perform a device search by confirming the **Scan for boxes** request or later by right-clicking on the found EtherCAT device and selecting **Scan** in the context menu as shown below:



- Confirm that the question to use online description and after a while you should be able to see the 9251 device in the project tree:



- To see the process data, please click on the 9251 in the project tree (a) and select the **Process Data** tab (b):

The screenshot shows the TwinCAT software interface. On the left, the Solution Explorer shows a project tree with a folder 'Devices' containing 'Device 2 (EtherCAT)'. Under 'Device 2 (EtherCAT)', there is a sub-folder 'Box 1 (9251 EtherCAT)' which is highlighted with a green box and labeled 'a'. The main window displays the 'Process Data' tab for the selected device, labeled 'b'. This tab contains several data tables:

Sync Manager:				PDO List:		
SM	Size	Type	Flags	Index	Size	Name
0	276	MbxOut		0x1A00	932.0	Transmit
1	276	MbxIn		0x1A01	292.0	Transmit
2	144	Outputs	F	0x1600	144.0	Receive
3	1224	Inputs	F			

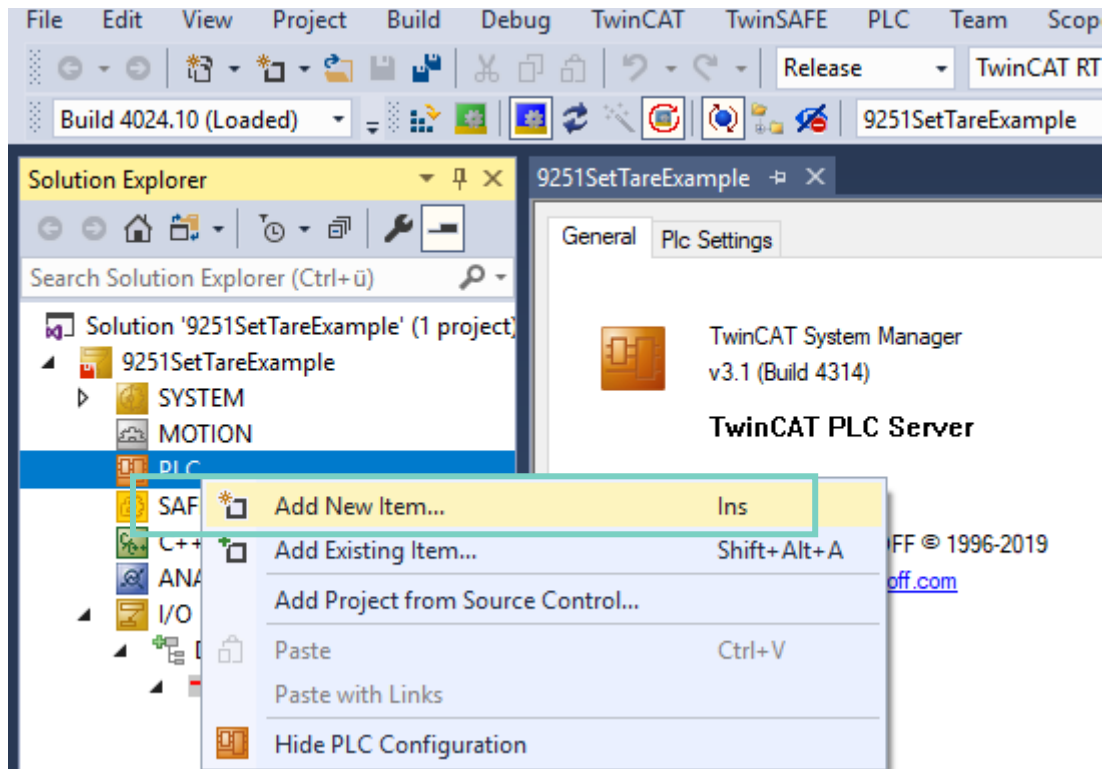
PDO Assignment (0x1C12):	PDO Content (0x1A00):		
<input checked="" type="checkbox"/> 0x1600	Index	Size	Offs
	0x2065:01	1.0	0.0
	0x2065:02	1.0	1.0

Name	Online	Type
9251module_STATUS1	0	USINT
9251module_STATUS2	0	USINT
9251module_MEAS_LIVE	2.5393963	REAL
9251module_CNT_LIVE	197	USINT
9251module_CNT_ARRAY	154	USINT
9251module_MEAS_ARR...	2.5302982	REAL

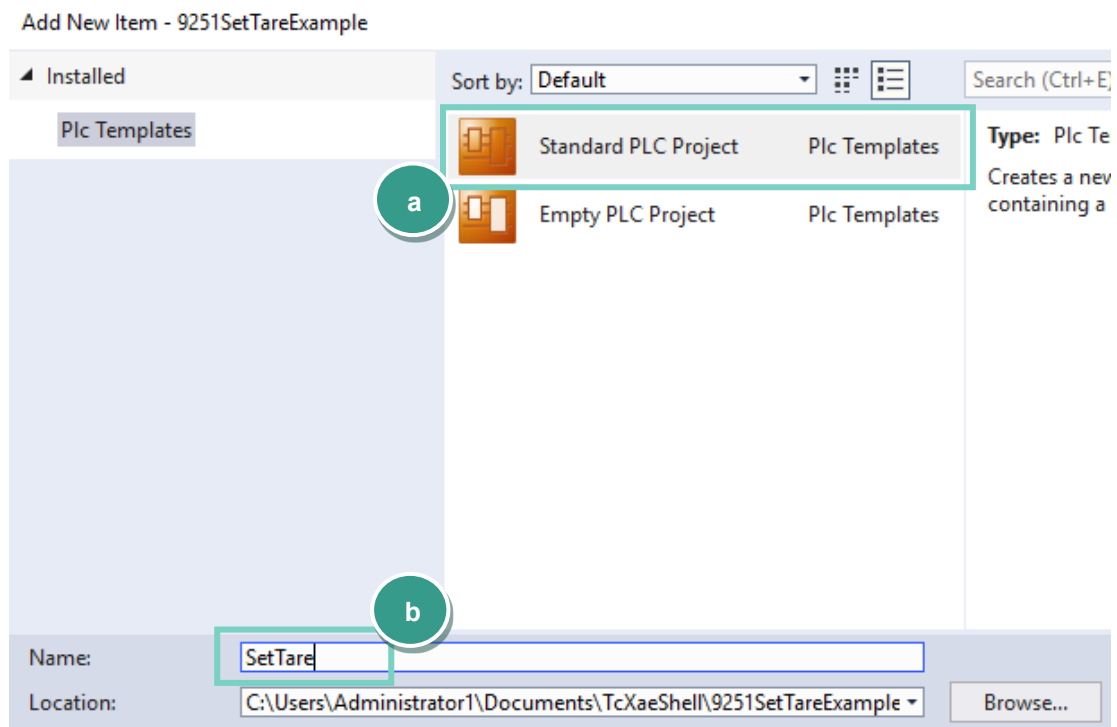
5 Create a sample program

In this section, you will learn how to create a simple PLC program to execute the tare function via PDO (Process Data Object). You will need to refer to section 7.3 *EtherCAT PDO – Process Data Objects* in 9251 documentation manual to understand the meaning of input bytes.

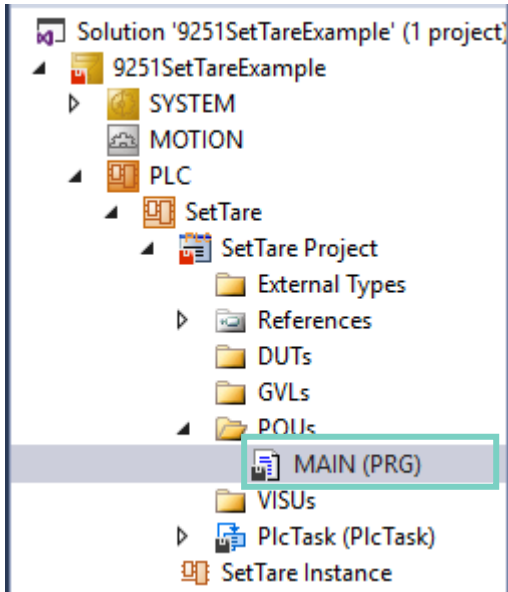
- Right-click **PLC** in the project tree and select **Add New Item...**



- Select **Standard PLC Project** (a) in the **Add New Item** dialog, enter **SetTare** as project name (b) and click **Add**



- Next, open the **MAIN (PRG)** file from **PLC** → **SetTare Project** → **POUs** with double click on it:



Example 1: Reading and Writing of PDOs

- Type in the following text in the **MAIN** block

```

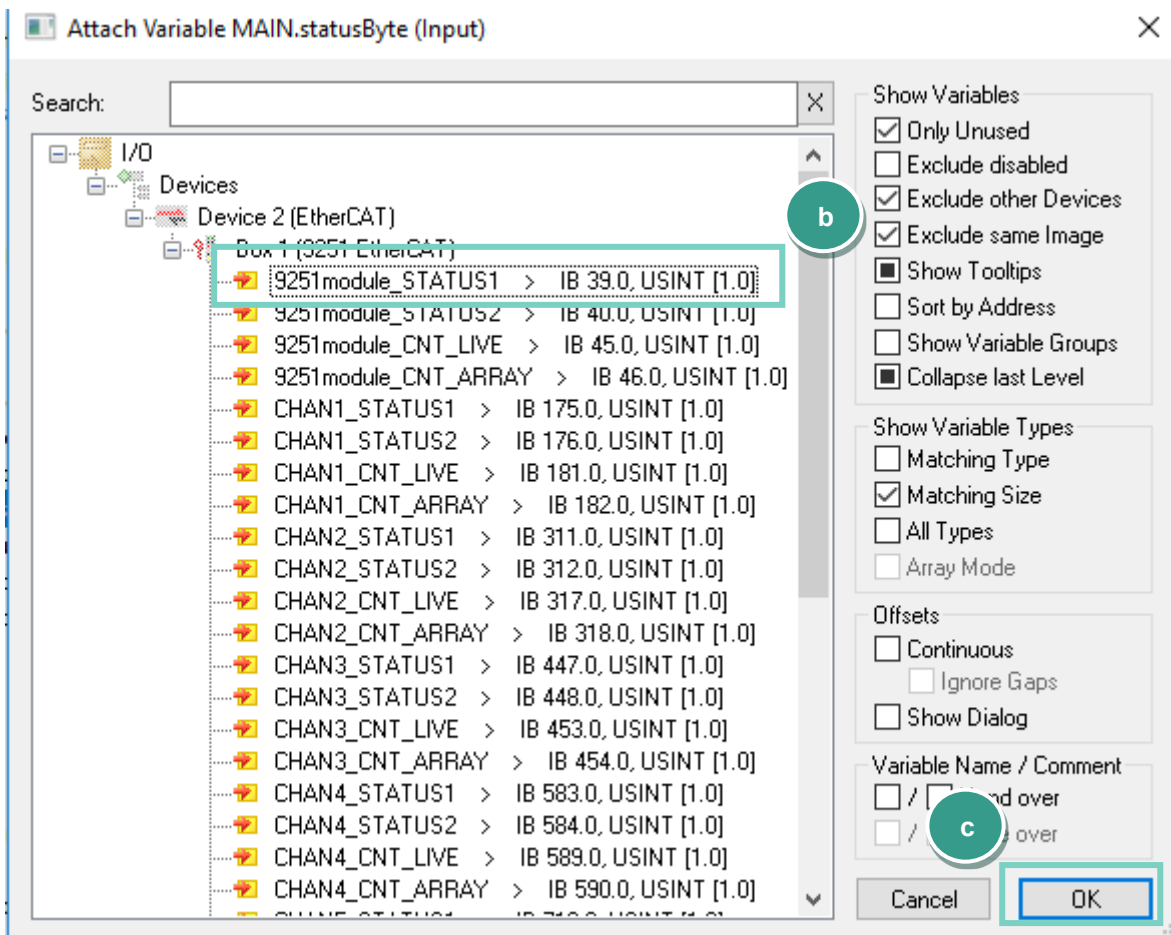
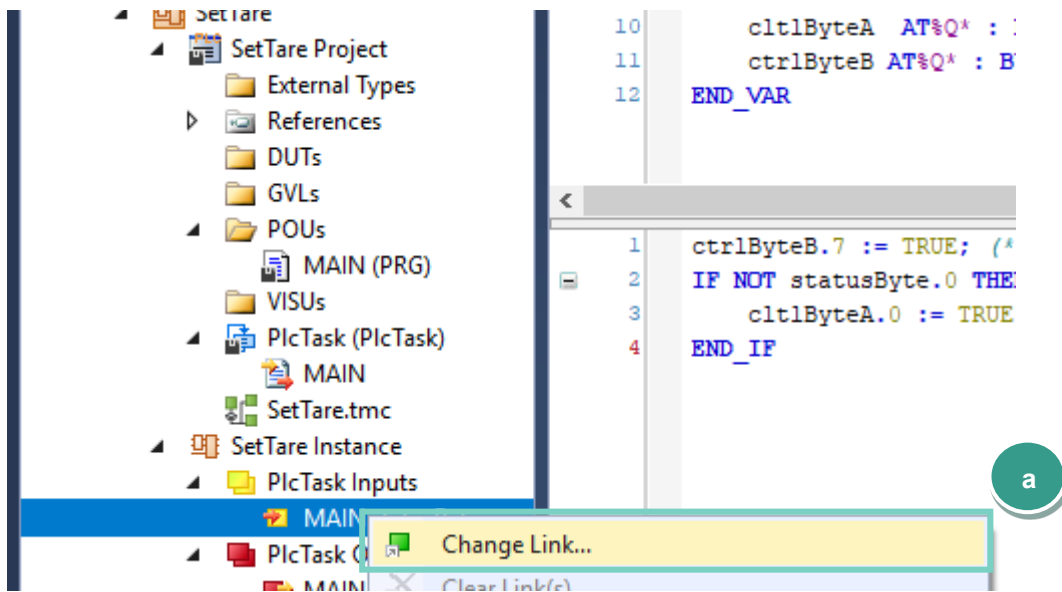
MAIN  ▸ × 9251SetTareExample
1  PROGRAM MAIN
2  VAR
3  END_VAR
4
5  VAR_INPUT
6      statusByte AT%I* : BYTE; // will be assigned to STATUS1
7  END_VAR
8
9  VAR_OUTPUT
10     ctrlByteA AT%Q* : BYTE; // will be assigned to CONTROL BYTE 1
11     ctrlByteB AT%Q* : BYTE; // will be assigned to CONTROL BYTE 2
12 END_VAR

1  ctrlByteB.7 := TRUE; (* PLC Control*)
2  IF NOT statusByte.0 THEN (* if tare is not active *)
3      ctrlByteA.0 := TRUE; (*Set TARE bit*)
4  END_IF
    
```

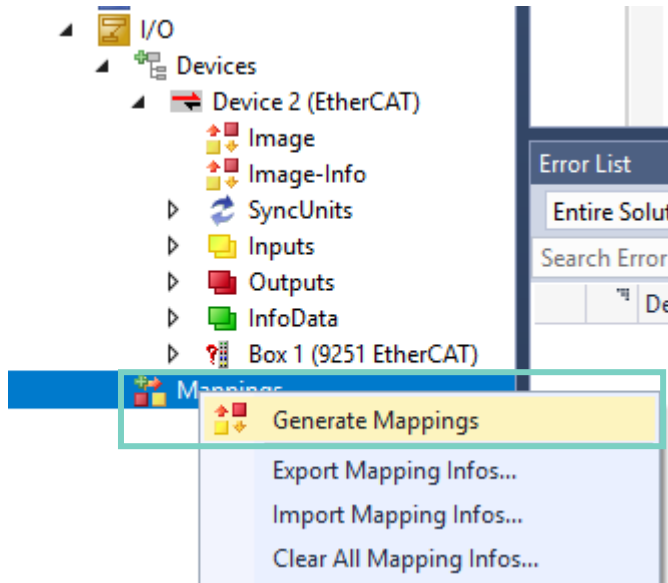
- Assign the input and output variables to the corresponded PDOs with the right-click on a variable and select **Change Link...**(a) from the context menu, select a corresponded PDO (b) and click OK (c):

Assignment:

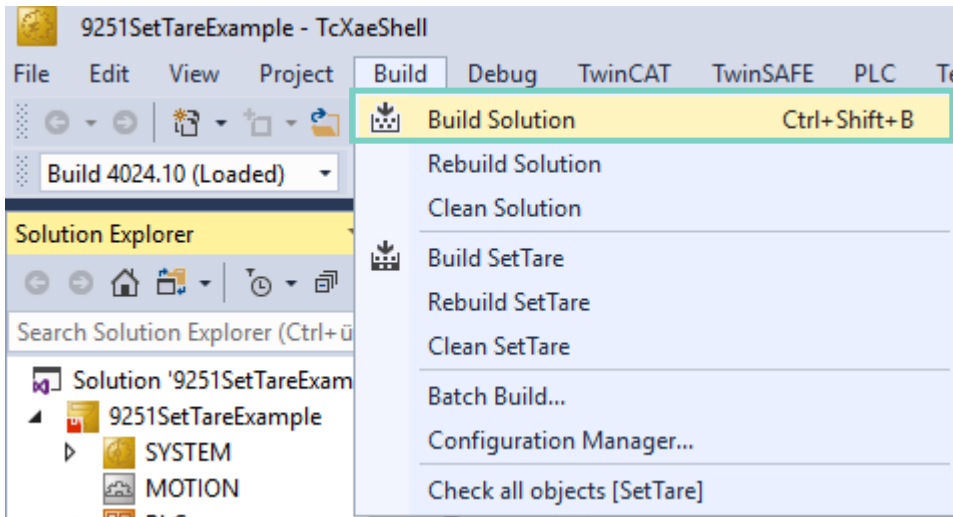
MAIN.statusByte - > *9251module_STATUS1*
MAIN.ctrlByteA → *9251module_CTRL1ch*
MAIN.ctrlByteB → *9251module_CTRL2ch*



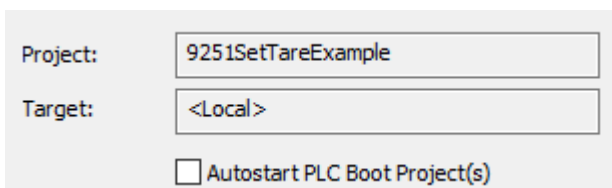
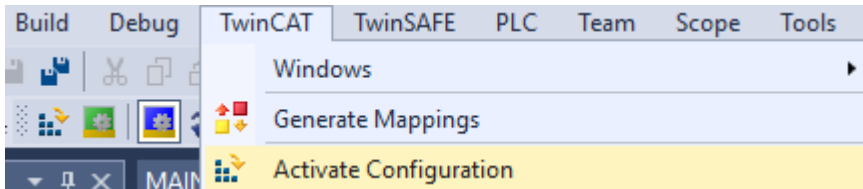
- Right-click Mappings → Generate Mapping:



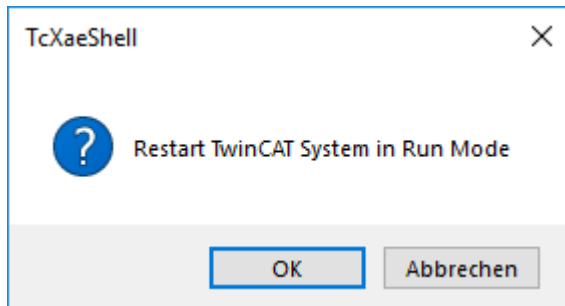
- Goto **Build** → **Build Solution** to build the project:



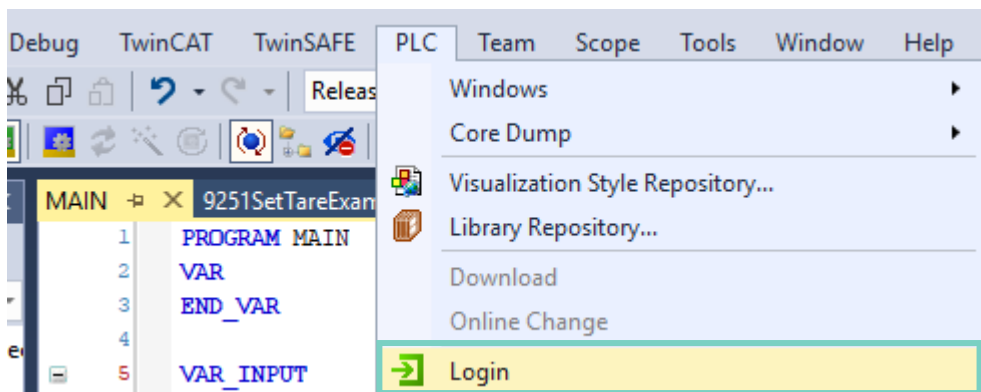
- Activate configuration via TwinCAT → Activate Configuration



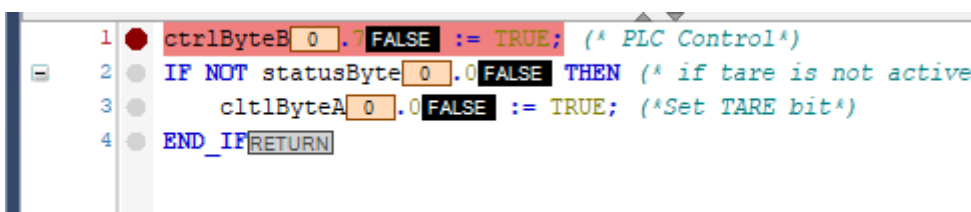
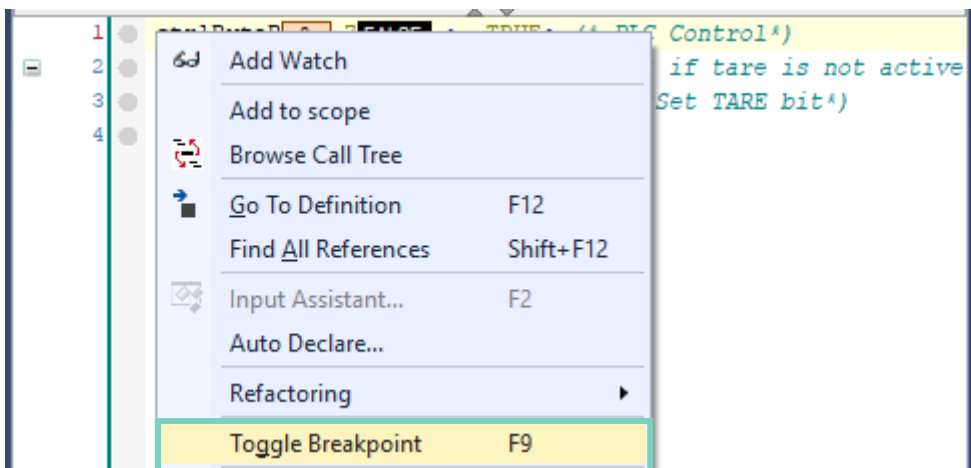
- Confirm starting in **Run Mode**:



- Goto **PLC** → **Login**: and if asked, confirm that program should be downloaded into the controller



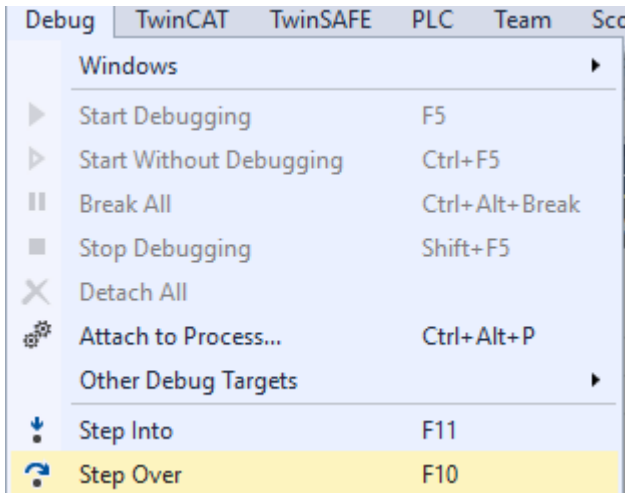
- Set a breakpoint (F9 or right-click → Toggle Breakpoint) to the first line to control the program execution step by step:



- Press the F5 key or click on the green start symbol to start the program execution:



- Execute the program line by line with the key **F10** oder via **Debug** → **Step over**



- When you reach the end of the IF condition (last line) the 7th bit in the control byte B (PLC control) as well as the 1st bit in the control byte A (Tare) should be set:

```

1  ctrlByteB[128].7 TRUE := TRUE; (* PLC Control*)
2  IF NOT statusByte[1].0 TRUE THEN (* if tare is not active *)
3      ctrlByteA[1].0 TRUE := TRUE; (*Set TARE bit*)
4  END_IF RETURN
    
```

- Double click the on the 9251 device in the project tree (a) to control the results. The `_STATUS1` byte (b) has a value of 1 (tare is active) and the current measurement value (c) is about zero

The image shows the TwinCAT project tree on the left and a variable declaration table on the right. In the project tree, 'Box 1 (9251 EtherCAT)' is selected and labeled with a red circle 'a'. The variable declaration table on the right shows the following data:

Name	Online	Type
9251module_STATUS1	X 1	USINT
9251module_STATUS2	0	USINT
9251module_MEAS_LIVE	0.03711772	REAL
9251module_CNT_LIVE	153	REAL
9251module_CNT_ARRAY	26	REAL
9251module_MEAS_ARRAY_00	0.046215773	REAL
9251module_MEAS_ARRAY_01	0.00075411797	REAL
9251module_MEAS_ARRAY_02	-0.0053017139	REAL
9251module_MEAS_ARRAY_03	-0.0053017139	REAL
9251module_MEAS_ARRAY_04	0.040150404	REAL
9251module_MEAS_ARRAY_05	0.00075411797	REAL
9251module_MEAS_ARRAY_06	-0.0053017139	REAL
9251module_MEAS_ARRAY_07	-0.011367083	REAL

The value '1' for `_STATUS1` is circled in red and labeled 'b'. The value '0.03711772' for `_MEAS_LIVE` is circled in red and labeled 'c'.

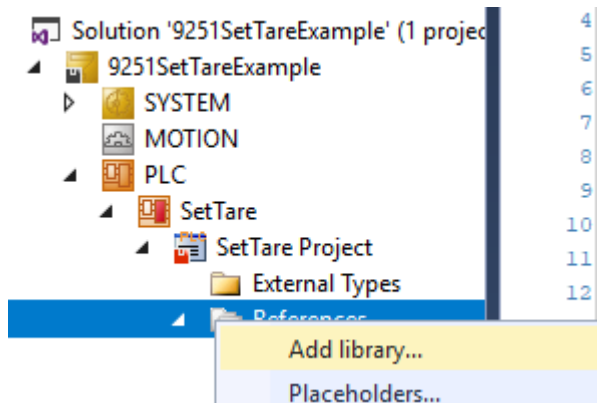
6 Further Examples

6.1 Read and Write of 'real' data types

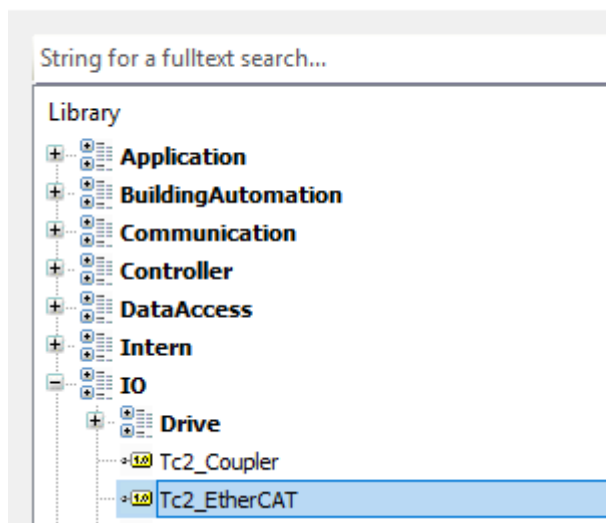
Example 2: Set and Get the Limit A – Lower Value

This example shows you how to write and read the Limit A – Lower Value.

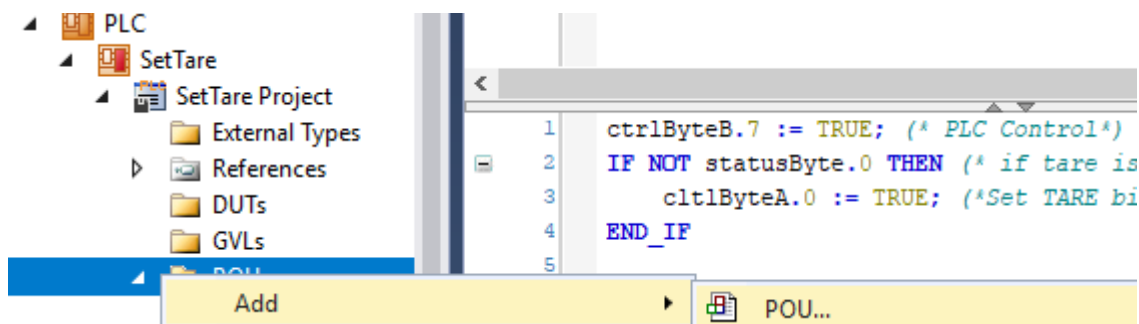
- Add the **Tc2_EtherCAT** library to your project to be able to use *FB_EcCoESdoRead* and *FB_EcCoESdoWrite* function blocks via **References** → **Add library**



Add Library



- Add a new **POU** (Program Organization Unit)



- Rename it to **WriteReadLimitALow** and click **OK**:

Add POU

Create a new POU (Program Organization Unit)

Name:
WriteReadLimitALow

Type

Program

Function Block

Extends: ...

Implements: ...

Final Abstract

Access specifier:

Method implementation language:
Structured Text (ST)

Function

Return type: ...

Implementation language:
Structured Text (ST)

Open Cancel

- Insert the call of the **WriteReadLimitALow** in the **MAIN** POU:

SetTare

- SetTare Project
 - External Types
 - References
 - DUTs
 - GVLs
 - POUs
 - MAIN (PRG)
 - WriteReadLimitALow (PRG)
 - VISUs

```
1 ctrlByteB.7 := TRUE; (* PLC
2 IF NOT statusByte.0 THEN (*
3   ctrlByteA.0 := TRUE; (*
4 END_IF
5
6 WriteReadLimitALow();
```

- Type in the following code into the created **WriteReadLimitALow** POU

Source code:

```

PROGRAM WriteReadLimitALow
VAR
    fbSdoWrite    : FB_EcCoESdoWrite;
    fbSdoRead     : FB_EcCoESdoRead;
    sNetId        : T_AmsNetId := '169.254.20.111.3.1'; // see note 1 below
    nSlaveAddr    : UINT := 1001;                       // see note 2 below
    nIndex        : WORD := 16#206A;                   // CoE Object - Limit A Lower Value
    nSubIndex     : BYTE := 0;                         // is always 0
    fLimitALow    : REAL := 1.43;                       // data to be written to 9251
    bExecute      : BOOL := TRUE;
    bError        : BOOL;
    nErrId        : UDINT;
END_VAR

fbSdoWrite(
    sNetId      := sNetId,
    nSlaveAddr  := nSlaveAddr,
    nIndex      := nIndex,
    nSubIndex   := nSubIndex,
    pSrcBuf     := ADR(fLimitALow),
    cbBufLen    := SIZEOF(fLimitALow),
    bExecute    := bExecute
);

IF NOT fbSdoWrite.bBusy THEN
    bExecute := FALSE;
    IF NOT bError THEN
        (* write successful *)
        bError := FALSE;
        nErrId := 0;
    ELSE
        (* write failed *)
        bError := fbSdoWrite.bError;
        nErrId := fbSdoWrite.nErrId;
    END_IF
END_IF

fbSdoWrite(bExecute := FALSE);

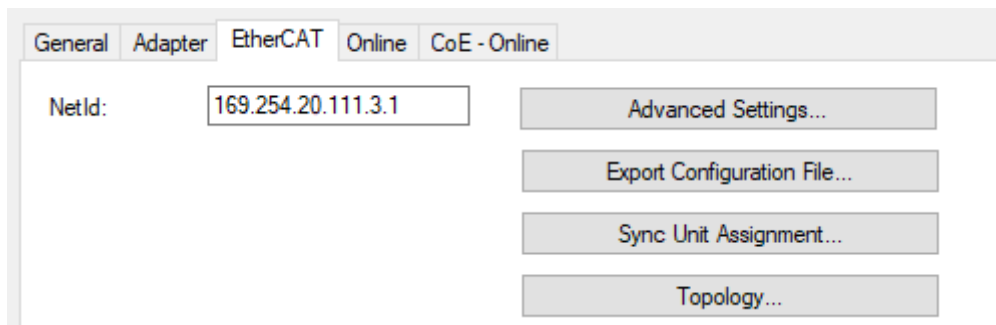
END_IF

fLimitALow := 0.0;

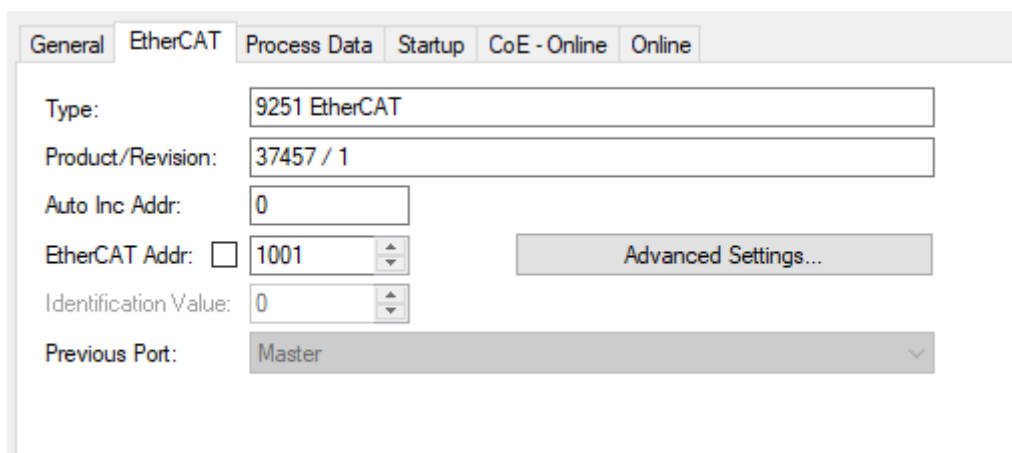
fbSdoRead(sNetId:= sNetId,nSlaveAddr :=nSlaveAddr, nIndex:=nIndex, nSubIndex :=nSubIndex,
pDstBuf:= ADR(fLimitALow), cbBufLen:=SIZEOF(fLimitALow), bExecute:=TRUE);
bError:=fbSdoRead.bError;
nErrId:=fbSdoRead.nErrId;


```

NOTE: You will find the **NetId** if you click your EtherCAT master device in the project tree and select the tab **EtherCAT**:



NOTE: You will find the EtherCAT slave address if you click the 9251 device in the project tree and select the tab **EtherCAT**:



- Build the project via **Build** → **Build Solution**, click on the **Login**  symbol and set a break point (F9) in the first code line:

```

1  ● fbSdoWrite (
2      sNetId '169.254.20' := sNetId '169.254.20',
3      nSlaveAddr 1001 := nSlaveAddr 1001,
4      nIndex 8298 := nIndex 8298,
5      nSubIndex 0 := nSubIndex 0,
6      pSrcBuf 16#C7A61060 := ADR(fLimitALow 1.43),
7      cbBufLen 4 := SIZEOF(fLimitALow 1.43),
8      bExecute TRUE := bExecute TRUE
9  );
10

```

- Start the program execution with the **F5** key or via **PLC** → **Start** and go step for step (**F10**) through the whole program until you reach the last line. Check if the written and read values are identical:

```

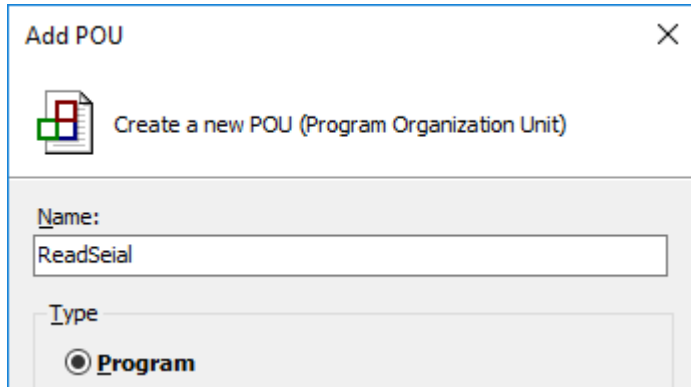
26 ● fLimitALow 1.43 := 0.0;
27
28 ● fbSdoRead(sNetId '169.254.20' := sNetId '169.254.20', nSlaveAddr 1001
29 ● bError FALSE := fbSdoRead.bError FALSE;
30 ● nErrId 0 := fbSdoRead.nErrId 0; RETURN

```

6.2 Read and Write of 'string' data types

Example 3: Read the serial number of 9251:

- Create a new POU as described above and name it **ReadSerial**:



- Write or copy the following source code into the new POU:

```

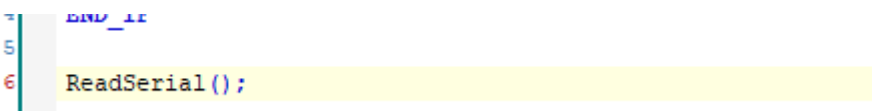
PROGRAM ReadSerial
VAR
    fbSdoRead : FB_EcCoESdoRead;
    sNetId    : T_AmsNetId := '169.254.20.111.3.1'; // see note 1 in the previous section
    nSlaveAddr : UINT := 1001; // see note 2 in the previous section
    nIndex     : WORD := 16#2073; // CoE Object – Serial number
    nSubIndex  : BYTE := 0; // is always 0
    abSerial   : STRING;
    bExecute   : BOOL := TRUE;
    bError     : BOOL;
    nErrId     : UDINT;
END_VAR

fbSdoRead(sNetId:= sNetId,
          nSlaveAddr :=nSlaveAddr,
          nIndex:=nIndex,
          nSubIndex :=nSubIndex,
          pDstBuf:= ADDR(abSerial),
          cbBufLen:=20,
          bExecute:=TRUE);

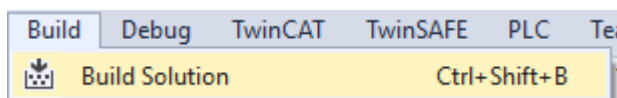
bError:=fbSdoRead.bError;
nErrId:=fbSdoRead.nErrId;

```

- Instert a call for the POU in the **MAIN** block:



- Build the Project via Build → Build Solution:



- Log in *PLC* → *Login*, set a break point to the last line and click *PLC* → *Start (F5)* to run the program

```
1  fbSdoRead(sNetId '169.254.20' := sNetId '169.254.20',
2      nSlaveAddr 1001 := nSlaveAddr 1001,
3      nIndex 8307 := nIndex 8307,
4      nSubIndex 0 := nSubIndex 0,
5      pDstBuf 16#C7A69AC8 := ADR(abSerial '12345678AB'),
6      cbBufLen 20 := 20,
7      bExecute TRUE := TRUE);
8  bError FALSE := fbSdoRead.bError FALSE;
9  nErrId 0 := fbSdoRead.nErrId 0; RETURN
```

- The serial number is read into the variable 'abSerial'